AIAA 98-4844

# CONCEPTUAL AIRCRAFT DESIGN ENVIRONMENT: CASE STUDY EVALUATION OF COMPUTING ARCHITECTURE TECHNOLOGIES

*Daniel Tejtel*[*]
*Dimitri N. Mavris*[†]
*Mark Hale*[‡]

School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150
www.asdl.gatech.edu

## Abstract

Designers need to use a variety of different codes in order to solve today's complex design problems; codes which must all be made to work together. Tools can be developed which facilitate the integration of these varied codes, so that they can be used together to solve a single problem. Using a computational architecture, a procedure has been set up which allows for a complete aerodynamic analysis of a High Speed Civil Transport. The computer architecture serves as a framework within which any number of diverse codes can be linked; data can be exchanged, stored, and otherwise managed; and decisions regarding the design of a vehicle can be made. The use of a computational tool called a Process Element as the method of code implementation allows for the basic analysis procedure to be easily modified and added to and to be used with higher-level, probabilistic-based design methods. By means of the High Speed Civil Transport aerodynamic analysis example problem described in this paper, the key features of the computational architecture, as well as its capabilities and limitations, are examined and evaluated.

## Introduction

A designer increasingly requires the guidance of probabilistic design techniques and simulation in order to completely understand complex design problems in aerospace engineering and arrive at viable and accurate solutions. Computing architectures facilitate this process by allowing a designer or designers to set up and a solve a variety of problems within a single design environment. Designers can link tools selected from an existing library of analyis tools, based on the nature of the problem at hand. It is advantageous for the architecture to possess the right tools, or even better, for the architecture to allow for the use of any tool, consequently allowing the designer to solve nearly any problem. This notion is a desired goal because a real-world implementation must consider that industry has accepted tools and requirements that are often used in place of those provided in a commercial system. In this case study, legacy tools are incorporated into a computing architecture in order to be coupled with newer design methods, such as Response Surface Methodology (RSM)[1] and Probabilistic Design[2] as shown below in Figure 1. These new design methods have been developed at Georgia Tech's Aerospace Systems Design Laboratory and applied to many current-day design problems. In this particular case, these tools are applied to the design-oriented analysis of a High Speed Civil Transport (HSCT) from an aerodynamic point of view.

Individual computer codes must be “compartmentalized" within a design framework so as to allow them to become building blocks which can then be linked with other such building blocks to construct the analysis process. These code blocks are known as agents. An agent is a specific computer code, a script, or scripts, which controls the program's execution, and a wrap which allows the code to interface with the architecture.[3] The roadmap

---

[*] Graduate Student, AIAA Student Member
[†] Assistant Professor, AIAA Senior Member
[‡] Research Engineer II, Corresponding Author, AIAA Professional Member

in Figure 1 demonstrates how standalone codes for different disciplinary analysis tools interact with the two probabilistic analysis tools.

The standalone codes are integrated as agents, which can be coupled with the RSM tool, shown in the figure as an RSE block, or with the probabilistic analysis tool, shown as an FPI block. The RSM tool can also be coupled with the probabilistic analysis tool. Each of the disciplinary blocks, aerodynamics, propulsion, structures, stability and control, and aerolasticity can be coupled with system-level analysis tools. The coupling of the aerodynamic tool with system level analysis will be described later on in the paper, as it and the propulsion analysis have already been implemented, but the other disciplinary analyses have not.

## Simulation

The simulation carried out using the selected tools is highly important. It is through this process that the designer can determine the key characteristics of the aircraft being designed, and can vary parts of the aircraft or stages of the design process to examine their effect on the overall concept. Accurate simulation allows the designer to consider a suite of alternatives and determine their effects on the entire system, without having to conduct any actual physical testing. In order to simulate the design process within an architecture, the codes must be linked together to perform the desired analysis. Current methods for linking computer codes usually results in tightly coupled systems because the implementation does not consider the complex information needed for decision-making by a designer. These systems prove to be restrictive for all but early conceptual design stages and, furthermore, these tightly linked codes may only be able to handle small amounts of data, and the linking may not lend itself to storing data as it is accumulated when numerous iterations of the design problem are performed. To extend the use of the computer-based simulation beyond the early phases of design, the various codes being used need to be integrated together in as modular a fashion as possible, and the integration links need to be highly flexible. Furthermore, the integration structure needs to have the capability of handling the potentially enormous amounts of information generated by the individual codes and passed back and forth between them, so that it may be eventually distilled by a designer in a useful form.

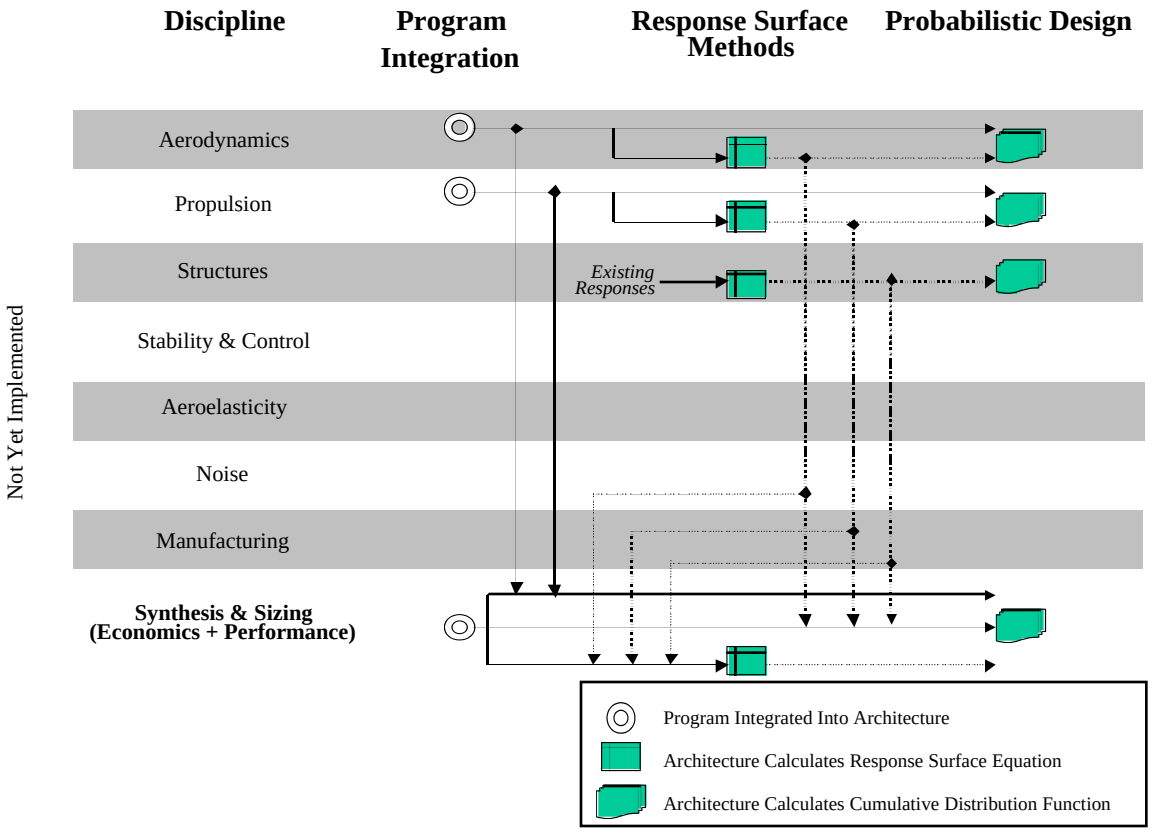


Figure 1. Stochastic Design Tool Roadmap

## Computing Architecture

The authors have formulated a new architecture technique called a Process Element (PE) that addresses the code integration and configuration issues described earlier.[4] A process element is a problem independent analysis object that contains program specific data (inputs and outputs), a link to the actual computer program, a script that specifies how to run the program, and a scope that dictates how the program is to be used intelligently within larger systems. When a process element is used within a computing environment, actual design data is linked to the process element to make its use problem specific. The use of process elements allows a majority of the work done by a designer to be focused on meta and actual design as opposed to configuration management. Therefore, a designer can spend more time solving the actual problem than preparing to solve it. Details about the inner workings of process elements will be highlighted in the case study described later in this paper.

IMAGE - Intelligent Multi-disciplinary Aircraft Generation Environment (IMAGE) - has been developed at the Georgia Tech Aerospace Systems Design Laboratory and is an architecture that provides a framework in which process elements are created and then combined into process models allowing for the problem to be executed.[5] Codes are selected by the designer as a function of the task which is to be accomplished. Using process element techniques deployed in the IMAGE architecture, the choice of codes is nearly unlimited. Legacy codes can be integrated along with shell scripts. There are essentially no platform restrictions; UNIX-based codes running on Sun or SGI workstations can be used alongside PC-based codes. The codes are executed using distributed computing, which allows them to be run on their native platforms, which may or may not be the machine on which the process model is being assembled. This allows different codes to be run concurrently and allows for faster execution. Furthermore, the use of process elements allows for significant flexibility and modularity. The process elements can be easily re-ordered to investigate the current design problem. Additionally, a designer needs to select only those process elements needed to analyze a specific requirement being investigated. As new codes are developed, they can be incorporated into the architecture without having to modify existing PE's.

## Case Study Implementation

A Conceptual Aircraft Design Environment (CADE) case study is documented here to highlight the strengths and weaknesses of process elements. The CADE case study is an implementation of analysis tools within the IMAGE environment. In turn, IMAGE is a deployment of process element technology believed to improve design. Through the configuration and execution of a problem within CADE, the issues of code integration and data manipulation can be examined.

### Problem Definition

A design-oriented analysis of an HSCT is the specific application that has been configured for this case study. The design of the HSCT is a topic which is investigated in the graduate design courses taught at Georgia Tech, AE6351 and AE6352. Within the scope of the two courses, design techniques are applied to disciplinary and system-level design and analysis of this revolutionary new vehicle. This case study focuses on a subset of this design problem: aerodynamic analysis. To solve this problem, several computer codes are utilized: a geometric modeler, several aerodynamic analysis codes, a mathematical code, and a synthesis code. The analysis process is shown in Figure 2.

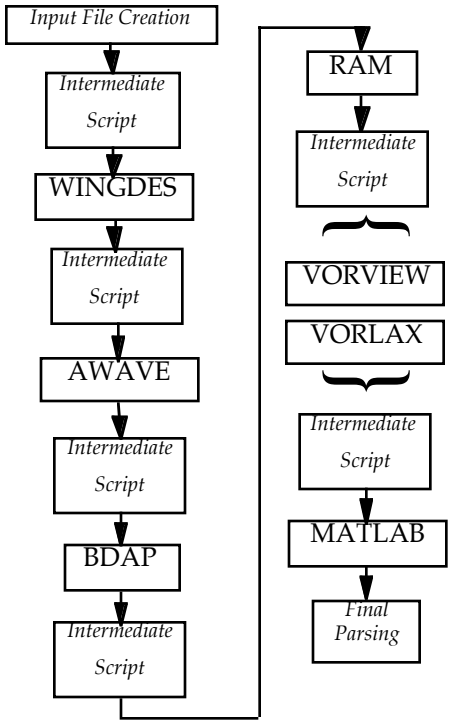


Figure 2: Aerodynamic Analysis Process

The entire analysis is based on a baseline HSCT configuration, developed in 1997 by the students in the AE6352 class and which was modified by this year's class as analyses were performed. Starting with this baseline geometry file, aerodynamic analyses are performed to determine the shape of the aircraft's drag polar as well as redefine, to a certain extent, its shape. Once enough data has been calculated so that the function for drag polar can be determined, the data is sent to the mathematics code. The mathematics code interpolates the points given to it and determines the drag polar equation coefficients $C_{Do}$, $K_{1,}$ and $K_2$, such that the drag equation developed is of the form:

$$C_D = C_{Do} + K_1 \cdot C_L + K_2 \cdot C_L^2 \qquad (1)$$

The resulting drag polar can then be sent to and used within the synthesis code, which will allow for higher-fidelity analysis of the entire aircraft.

## Legacy Code Implementation

Specific details of the codes and scripts used for the case study as well as their function and implementation within the problem are described below.

The process begins with a script which creates a master input file. The file is created by compiling the data describing the aircraft geometry and flight condition into namelist format. The values used for the file are drawn from a database which contains a complete data model for the aircraft. This data model can be used not only for the aerodynamic analysis used in this specific case study, but also for structural or other analyses of the HSCT. The existence of this global model helps ensure that designers of all disciplines will always be working from the same data set and will allow them to observe the effects of changes to a single design variable on the entire system. The design variables used in this global model are very general, so they must be converted into values which are more meaningful to a disciplinarian, and which can be used by disciplinary analysis tools. For example, for an aerodynamicist, non-dimensional coordinates for points which define the shape of a wing would need to be converted into more meaningful parameters, such as aspect ratio and taper ratio.

### WINGDES

The Wing Design program (WINGDES) is used to redefine and optimize the shape of the wing.[6] It provides optimum twist and camber for a given geometry and flight condition. The data used to create its input file comes from the global model, but has been modified so as to make it meaningful to WINGDES. When using WINGDES, the flexibility afforded the designer by the design architecture becomes apparent. The version of the program used in this case study was compiled for execution on an RS6000 platform. However, the entire process was run from an SGI workstation. The architecture allows the user to run any combination of codes, regardless of platform requirements. In the past, this same aerodynamic analysis had to be broken down into two big subsections as a function of the platform requirements for the codes. However, using the design architecture, process automation is maintained as the architecture simply runs WINGDES on a different machine and proceeds on with the next step of the process.

### BDAP

BDAP is a series of aerodynamic analysis codes developed by Boeing.[7] For this application, it is used only to determine skin friction drag on the aircraft. A script is used to assemble an input file using variables from the global data model. Once the program has run, the resulting output file is parsed to obtain the required drag values, which are stored for later use. BDAP also requires an RS6000 platform on which to run, a requirement which is taken care of automatically by the architecture without having to interrupt the analysis process.

### AWAVE

The wave drag analysis and design code (AWAVE) is used primarily to apply area ruling to the fuselage.[8] Given the nature of the HSCT, area ruling is essential to minimize wave drag through the transonic and supersonic regimes. Once again, an input file is created based on the converted design variable values calculated from the global data model as well as more detailed geometric information obtained from a baseline geometric definition of the HSCT. Running AWAVE results in fuselage dimensions optimized so as to minimize the fuselage-induced wave drag. In order to ensure that a realistic and usable fuselage will be produced, constraints have been imposed on minimum allowable fuselage diameter. AWAVE also requires an RS6000 machine for execution, a requirement which the architecture handles as it did for BDAP and WINGDES.

### RAM

Rapid Aircraft Modeler (RAM) is the geometric modeling tool used in this application.[9] A RAM process element has been configured within the architecture that allows the aircraft geometry to be automatically converted into a geometry file which can be sent to the aerodynamics code for analysis. This automatic file conversion is a result of the source code having been modified to eliminate user interaction and automate the process. Such source code modifications are sometimes necessary in cases where user input is required, but the input does not differ from one case to the next. RAM is an extremely useful tool but its implementation requires a significant amount of work. The amount of data which is contained in a typical RAM file and which can be extracted for use in other parts of the process can be enormous, on the order of 200 parameters for a complete configuration model. Consequently, the data structures and scripts associated with the RAM agent are themselves quite large, and although fairly simple, require a significant amount of time to implement and disk space to store. In this particular application, a script had to be written to piece together a RAM file based on a baseline configuration and modified wing and fuselage parameters.

### VORLAX

VORLAX is the aerodynamics code used for this application.[10] It is based on a vortex lattice method. The VORVIEW graphical front end to VORLAX is also used, and has been modified to aid in the automation of the process. VORLAX requires two files as input. The first file is the geometry file generated by RAM. This file can be transferred directly to VORLAX, without any restructuring or other manipulation of the data contained within. The second file is a case file which contains various control parameters as well as the cases (Mach Number and Angle-of-Attack sweep) for which to run the analysis. This file must be generated by the user, and is done by a script which is run earlier. VORLAX also requires user input to generate the analysis grid. In order to maintain user input capability but retain complete automation, a script has been written which generates a file containing the grid parameters as defined by the user early in the process. With this implementation, the user can modify the grid parameters before running the analysis. Once the process has begun to run, the grid parameter file is automatically generated and the process can continue uninterrupted. This type of automation is possible because although the grid generation parameters may change from aircraft to aircraft, they will be consistent for all cases involving the same aircraft. As a result of its execution, VORLAX generates various aerodynamic coefficients which may be used as input to another code. In this case, induced drag terms are compiled as a function of different lift coefficients and this data, along with the $C_{D0}$ term determined previously, is sent to MATLAB to be interpolated.

### MATLAB

MATLAB is used to generate the coefficients for the actual drag polars.[11] It takes the output data from VORLAX and BDAP/AWAVE and performs a quadratic fit of the data. A script has been written which takes the lift and drag data and assembles a MATLAB "m" file which contains a matrix of the lift coefficients and their associated drag coefficients, the command to perform the polynomial fit, and a command to output the resulting data, the $C_{Do}$, $K_1$, and $K_2$ coefficients, to a file. Once MATLAB has been executed and has determined the coefficients of the quadratic drag polar, the last step in the analysis process is to parse the newly created file and store the data.

### FLOPS

Flight Optimization System (FLOPS) is the synthesis code of choice used in this case study.[12] FLOPS is representative of typical FORTRAN programs which use namelist input. Namelist input simply means that variable names and values are placed in a text file which is read in by a program. In this case study, FLOPS is used to perform system-level analyses of the HSCT, which deal with vehicle sizing and economic analysis. The FLOPS source code has been modified to accept and use the design variables used as the baseline definition for the aircraft and defined in the global data model. Additionally, FLOPS has been modified to accept the drag polars created in the steps above as aerodynamic input. This capability allows for higher fidelity results to be obtained from the FLOPS analysis as the aerodynamic data it is using is much more representative of the aerodynamic qualities of the actual aircraft. Even though FLOPS has the capability to use the new drag polars, it retains the ability to use any of its other aerodynamic analysis capabilities. Because of its namelist input format,

parameters can be easily and quickly changed to fit the designer's needs. Since FLOPS has been modified to use the same basic variable set as the baseline definition for the aircraft as is used to generate the aerodynamic data, changes to the geometry definition can be easily made and their impact on the entire system assessed. This general data model is not used only for aerodynamic data however, FLOPS has also been modified to calculate structural component weights as a function of these variables, increasing yet again the fidelity of its results.

The implementation of the design process within the computational architecture is shown in Figure 3. It is important to note that BDAP, AWAVE, and WINGDES are all executed within a single script, which is represented by the `aa_98` icon. The execution of this design process within the computation architecture is shown in Figure 4. The VORLAX-gridded HSCT is displayed as it is analyzed.

## Stochastic Design

The two techniques used to implement stochastic design principles are Response Surface Methods (RSM) and Probabilistic Design. Both of these techniques allow the disciplinary analysis described above to be used for big picture-type analyses. RSM consists of approximating complex analyses with second order polynomial equations which are functions of selected design variables. To obtain these equations, a Design of Experiment table is used to select and run test cases and the results of these cases

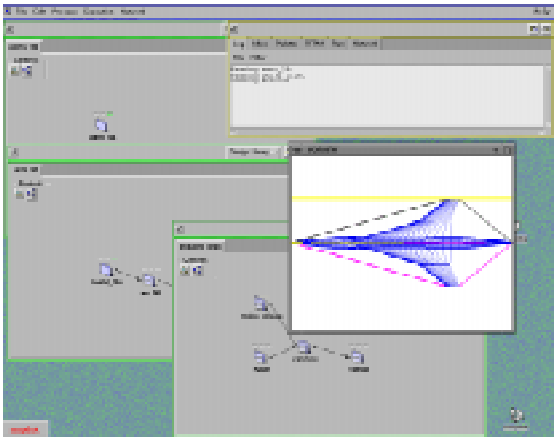

Figure 4: Aerodynamic Analysis Process Execution

are regressed to obtain the coefficients for the polynomial approximation. FPI consists of running a few selected trial cases to approximate Cumulative Distribution Functions (CDF's).

## Response Surface Methodology

To generate response surface equations (RSE's), a number of analysis cases ranging from a few dozen to hundreds must be run. In essence, whatever the analysis process is, it must be repeated as often as necessary, and all the relevant results stored. This repetition can become quite tedious since for every case, the values of the selected design variables must be changed according to the DoE, all resulting input files changed accordingly, and the results stored in such a way that they can be matched up with the input conditions. Rather than manually changing

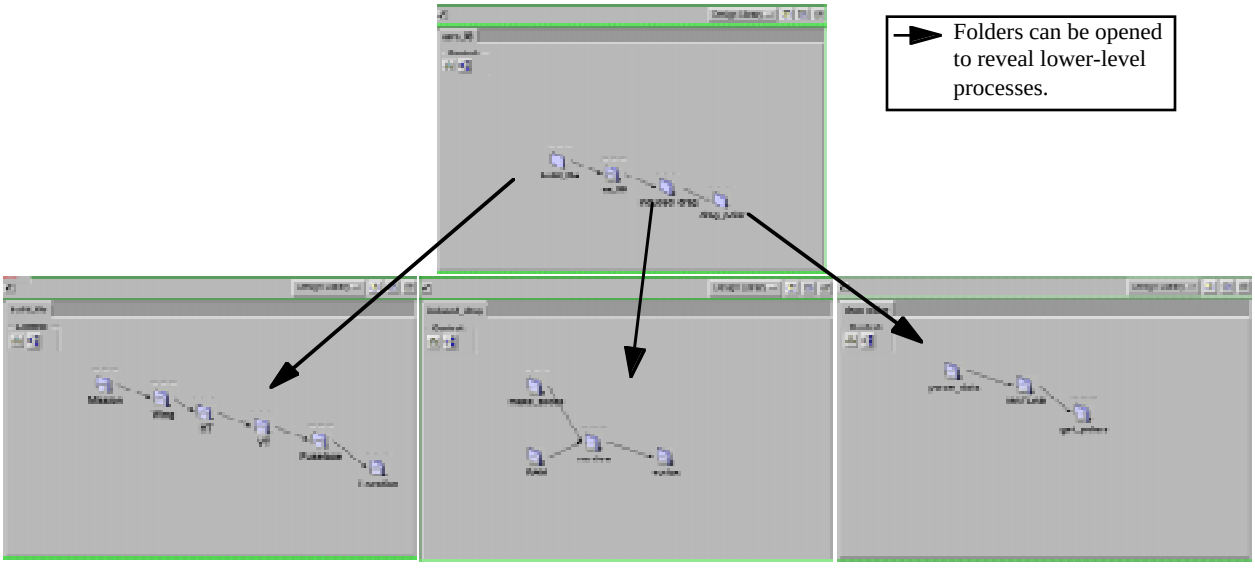


Figure 3: Aerodynamic Analysis Process Plan

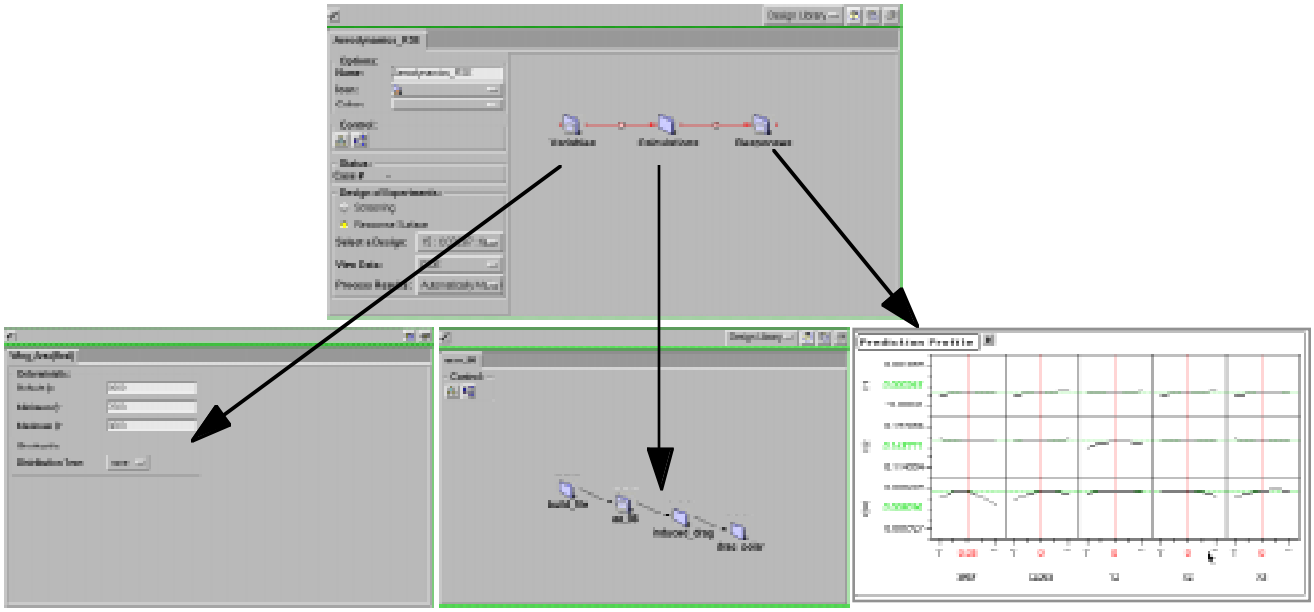

Figure 5: Response Surface Module

each of the design variables before every case, the architecture allows the designer to specify the design variables, DoE, and responses ahead of time, as shown in Figure 5.

Any type of analysis module can be used as the basis for generating an RSE. Using the architecture's plug-and-play capabilities, any existing process element or combination of process elements can be dragged into the RSE generation environment and used to generate the necessary responses. With the design variables and DoE selected, the architecture then performs all of the required runs, modifying the input file and storing the results for each iteration. This frees the designer from having to sit in front of the computer for each of the runs. With this "single-click" execution, the designer is free to either work on other projects or can let the problem run overnight. Additionally, since the architecture allows the designer to select the specific values to save, data storage requirements can be as high or as low as desired. No intermediate information needs to be saved, unless the designer specifically wants it to be. Historically, once the test case results had been compiled, they had to be exported to a different machine so that the regression calculations could be performed. Using the architecture, a code that performs the regression as part of the overall process can be added on, therefore maintaining complete automation of the process. In this implementation, MATLAB is used once again to perform these calculations. Prediction profiles generated as a result of a sample case executed using the architecture are shown in Figure 5. This figure shows the sensitivities of the three drag polar coefficients to wing area (SREF), design lift coefficient (CLDES), and three non-dimensional wing geometry design variables (Y2, X2, X3).

Once the RSE coefficients have been determined, they can be used within FLOPS to give an even more accurate aerodynamic model of the aircraft. As mentioned before, FLOPS has been modified to accept any single drag polar as input, which will yield significantly more accurate lift and drag estimates for each configuration being considered. However, with the RSE implementation, the lift and drag data now become a function of the design variables, which means that the drag polar will change as the aircraft geometry changes. This flexibility is achieved by storing sets of RSE coefficients for different mach numbers and altitudes and allowing FLOPS, by means of variable settings in the input file, to select the appropriate sets of coefficients to calculate a new drag polar for each individual configuration being analyzed by FLOPS.

## Probabilistic Design

The other stochastic design tool is Probabilistic Design. To carry out Probabilistic Design, Fast Probability Integration (FPI) is used. FPI is a probability-based tool which allows for the approximation of cumulative distribution functions. These approximations can be determined using a relative handful of trial cases, as opposed to the thousands necessary to determine a complete CDF. Once again, iteration is a key part of the process

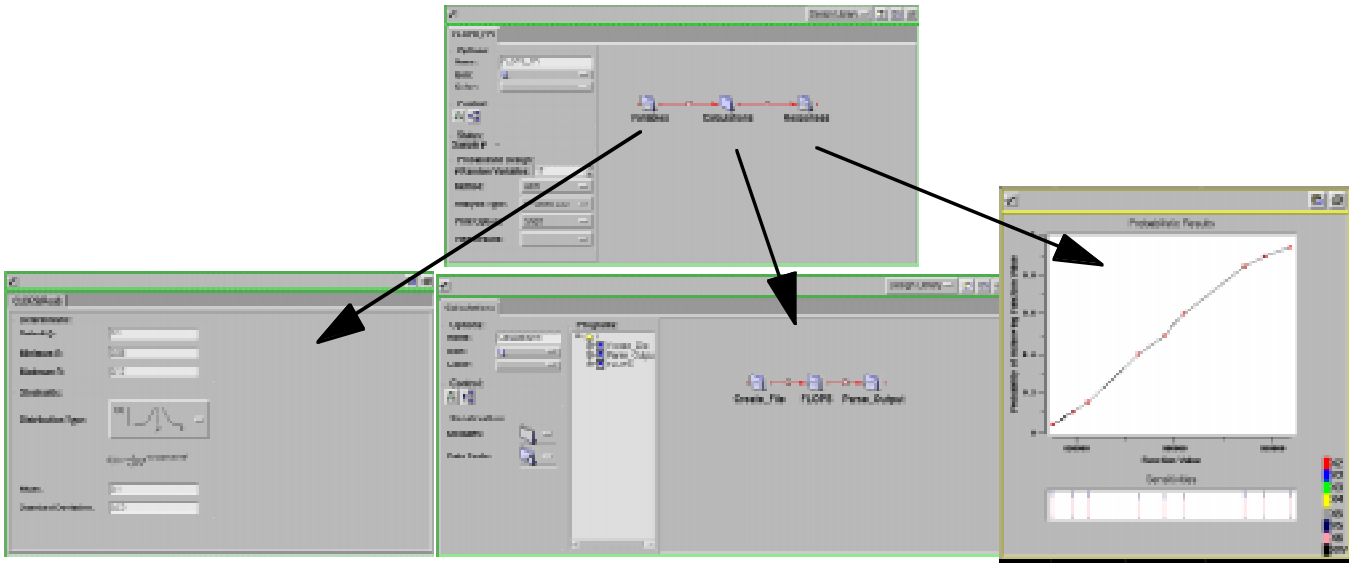

Figure 6: Probabilistic Design

when using FPI, since whatever code is being used to generate the results must be run again and again as the design variables are perturbed to obtain the different input values used to generate the CDF. The architecture once again handles the iteration, once the designer has selected what type of approximation is desired and has configured the process, as shown in Figure 6.

In the past, the FPI results had to exported to another program, such as EXCEL, so that they could be plotted and a graphical representation of the CDF obtained. Using the computational architecture, the data generated by FPI can be passed on to another program, and the CDF curved generated as part of the process. Such a curve is shown in Figure 6.

The CDF displayed in Figure 6 is a result generated by FLOPS, into which the previously generated RSE's have been inserted. The CDF curve shown above is an approximation of an actual CDF. It displays the probabilities associated with achieving a certain gross weight for the HSCT.

Using the architecture and MATLAB, the FPI results can be stored and the CDF automatically generated without the process having to be stopped to transfer the data. This capability gives the designer freedom to quickly assess the results of a current case, make changes if necessary, and run new cases without having to change computers or even change codes. As in the case of the RSE implementation, the code used to generate the FPI results can be any code or combination of codes already implemented as process elements within the architecture, or a new code can be added and used.

## Data Exchange

Information handling throughout the process is an important issue, especially when several different codes are being made to run in succession and if iteration is being performed. In this case, the architecture handles data exchange between programs as well as global storage of user defined design variables. Translators, developed as shell scripts, are used when necessary, as in the case of the global design variable to program variable exchange, to make sure that variables get transformed properly, or that the output data of one code becomes valid input data for another. All data generated during program execution can be viewed during the actual execution, or can be stored for later review. The designer can choose to store all the data generated during an analysis case, or simply the input variables and end results. This is useful for design studies, probabilistic analysis, and architecture. In the case of design studies, it allows the designer to view the effects of configuration-type changes on various aspects of the aircraft. In terms of probabilistic analysis, FPI can be used to determine the likelihood of achieving a certain wave drag value, or an overall complete drag value. Since data is stored sequentially as it is generated, the designer can easily view the progression of a result as values are changed, or in the case of multiple iteration-type operations, can compare input values with their results. This capability is useful if the designer wishes to re-run an individual case, which could occur if one case yielded unexpected or completely divergent results.

## Conclusion

The case study outlined in this paper demonstrates the potential for the automation of advanced stages of the design process through the use of process elements. Using the IMAGE architecture, individual analysis or design codes can be selected by a designer and linked together as necessary. Since most of these codes are legacy codes, they require pre- and post-processors, which are created in the form of simple shell scripts. New codes can be added at any time without changing the existing ones, and of those that are already implemented, designers can pick only those that are needed to solve the current design problem. Furthermore, all these disciplinary tools can then be used within the scope of higher level analysis tools, such as the Response Surface Methodology and Fast Probability Integration tools described above. The designer is also free to use codes located on remote machines, or which are specific to different platforms. Furthermore, the automation and linkage capabilities afforded by the architecture, along with its information handling capabilities, make it an essential tool for design studies.

## Acknowledgments

The authors acknowledge Mr. Samuel Dollyhigh of NASA Langley-Systems Analysis Branch, coordinator of the NASA MDA Fellowship Program for continuing support of this research. The authors would also like to acknowledge the assistance of Mr. Songtao Qiu and Mr. Nathan Hines, whose intimate knowledge of every aspect of the aerodynamic analysis proved to be essential during the integration of the process into the architecture.

## References

[1] DeLaurentis, D.A., Mavris D.N., Schrage D.P., "System Synthesis in Preliminary Aircraft Design Using Statistical Methods,' 20th International Council of the Aeronautical Sciences (ICAS) Congress, Sorrento, Italy, September 8-13, 1996.

[2] Southwest Research Institute, FPI User's and Theoretical Manual, San Antonio, TX, 1995.

[3] Hale, M.A. and Craig, J.I., "Use of Agents to Implement an Integrated Computing Environment," Computing in Aerospace 10, AIAA, San Antonio, TX, March 28-30, 1995. AIAA-95-1001.

[4] El Aichaoui, S.,Hale M.A., Craig J.I., "Building Design Applications Using Process Elements," 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St.Louis, MO, September 2-4, 1998.

[5] Hale, M. A., "An Open Computing Infrastructure that Facilitates Integrated Product and Process Development from a Decision-Based Perspective," Doctoral Dissertation, Georgia Institute of Technology, School of Aerospace Engineering, July, 1996.

[6] Carlson, H.W. and M.J. Mann, "Survey and Analysis of Research on Supersonic Drag-Due-to-Lift Minimization With Recommendations for Wing Design," NASA TP-3202, 1992.

[7] Middleton, W.D. and J.L. Lundry, "A System for Aerodynamic Design and Analysis of Supersonic Aircraft," NASA CR-3351, 1980.

[8] "AWAVE User's Guide for the Revised Wave Drag Analysis Program," NASA Langley Research Center, September, 1992.

[9] Gelhaussen, P., "Rapid Aircraft Modeler", Informal Briefing, NASA Ames Research Center, 1996.

[10] Miranda, L.R., R.D. Elliott, and W.M. Baker, "A Generalized Vortex Lattice Method for Subsonic and Supersonic Flow Applications," NASA CR-2865, 1977.

[11] The MathWorks, Inc., "On-Line MATLAB User's Guide, Version 5.0," www.mathworks.com, Copyright 1997.

[12] McCullers, L. A., "Flight Optimization System User's Guide, Release 5.85," NASA Langley Research Center, Revised 24 October 1995.